下载APP

04 | AOF日志：宕机了，Redis如何避免数据丢失？

2020-08-12 蒋德钧

Redis核心技术与实战

进入课程 >

讲述：蒋德钧
时长 14:40　大小 13.45M

你好，我是蒋德钧。

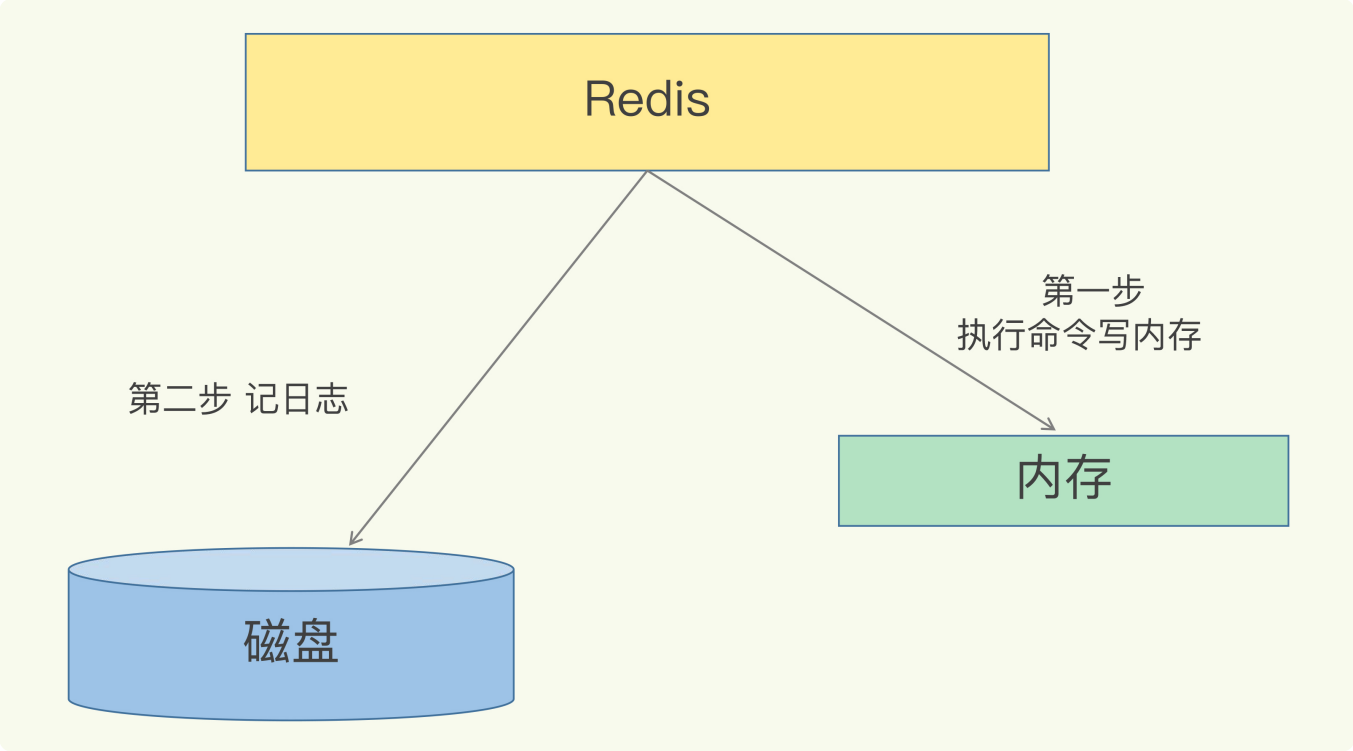
如果有人问你：“你会把 Redis 用在什么业务场景下？”我想你大概率会说：“我会把它当作缓存使用，因为它把后端数据库中的数据存储在内存中，然后直接从内存中读取数据，响应速度会非常快。”没错，这确实是 Redis 的一个普遍使用场景，但是，这里也有一个绝对不能忽略的问题：**一旦服务器宕机，内存中的数据将全部丢失。**

我们很容易想到的一个解决方案是，从后端数据库恢复这些数据，但这种方式存在两个问题：一是，需要频繁访问数据库，会给数据库带来巨大的压力；二是，这些数据是从慢速数据库中读取出来的，性能肯定比不上从 Redis 中读取，导致使用这些数据的应用程序响应变慢。所以，对 Redis 来说，实现数据的持久化，避免从后端数据库中进行恢复，是至关重要的。

目前，Redis 的持久化主要有两大机制，即 AOF 日志和 RDB 快照。在接下来的两节课里，我们就分别学习一下吧。这节课，我们先重点学习下 AOF 日志。

AOF 日志是如何实现的？

说到日志，我们比较熟悉的是数据库的写前日志（Write Ahead Log, WAL），也就是说，在实际写数据前，先把修改的数据记到日志文件中，以便故障时进行恢复。不过，AOF 日志正好相反，它是写后日志，“写后”的意思是 Redis 是先执行命令，把数据写入内存，然后才记录日志，如下图所示：

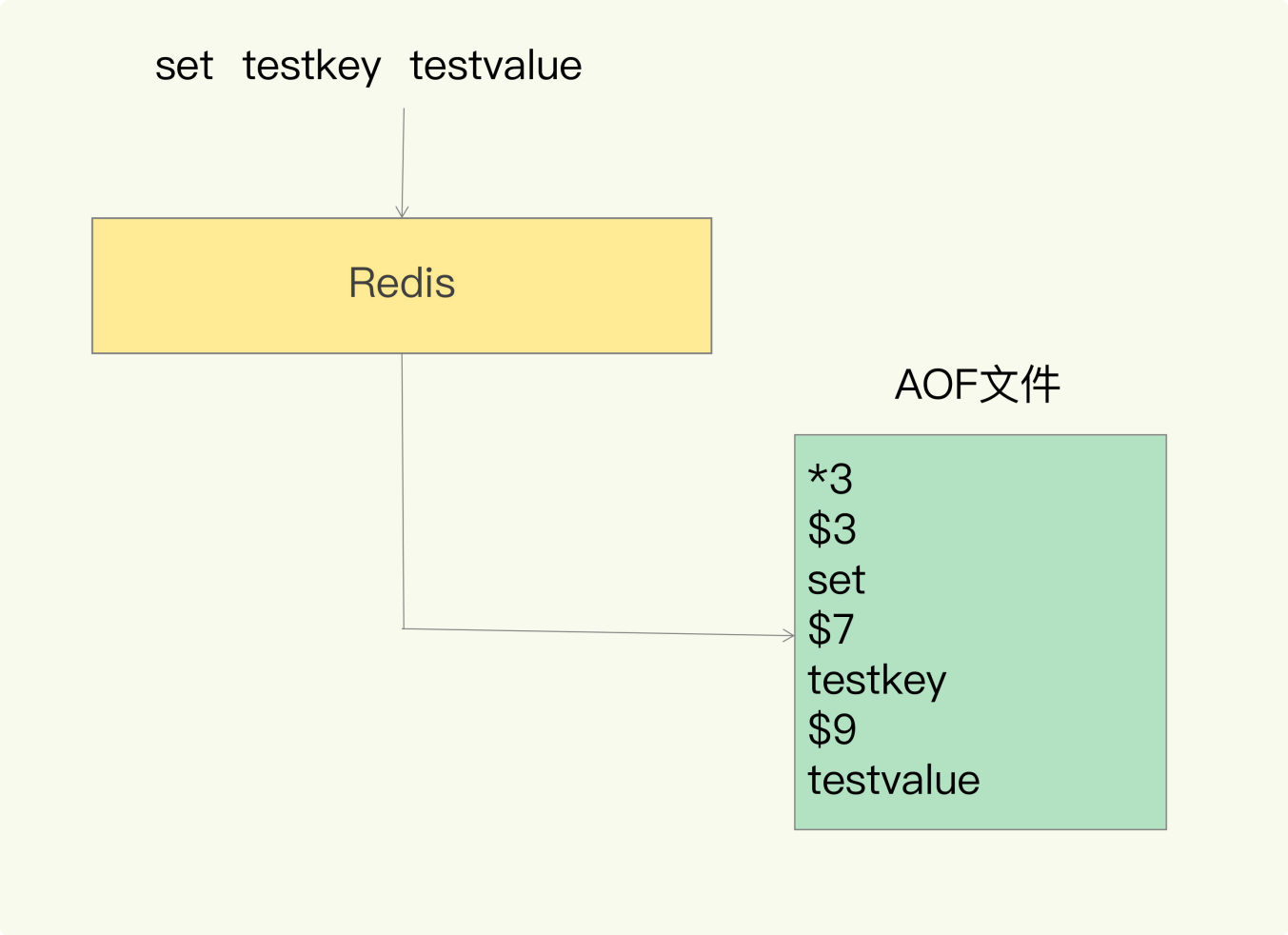


Redis AOF操作过程

那 AOF 为什么要先执行命令再记日志呢？要回答这个问题，我们要先知道 AOF 里记录了什么内容。

传统数据库的日志，例如 redo log（重做日志），记录的是修改后的数据，而 AOF 里记录的是 Redis 收到的每一条命令，这些命令是以文本形式保存的。

我们以 Redis 收到“set testkey testvalue”命令后记录的日志为例，看看 AOF 日志的内容。其中，“*3”表示当前命令有三个部分，每部分都是由“$+数字”开头，后面紧跟着具体的命令、键或值。这里，“数字”表示这部分中的命令、键或值一共有多少字节。例如，“$3 set”表示这部分有 3 个字节，也就是“set”命令。



Redis AOF日志内容

但是，为了避免额外的检查开销，Redis 在向 AOF 里面记录日志的时候，并不会先去对这些命令进行语法检查。所以，如果先记日志再执行命令的话，日志中就有可能记录了错误的命令，Redis 在使用日志恢复数据时，就可能会出错。

而写后日志这种方式，就是先让系统执行命令，只有命令能执行成功，才会被记录到日志中，否则，系统就会直接向客户端报错。所以，Redis 使用写后日志这一方式的一大好处是，可以避免出现记录错误命令的情况。

除此之外，AOF 还有一个好处：它是在命令执行后才记录日志，所以**不会阻塞当前的写操作**。

不过，AOF 也有两个潜在的风险。

首先，如果刚执行完一个命令，还没有来得及记日志就宕机了，那么这个命令和相应的数据就有丢失的风险。如果此时 Redis 是用作缓存，还可以从后端数据库重新读入数据进行

恢复，但是，如果 Redis 是直接用作数据库的话，此时，因为命令没有记入日志，所以就无法用日志进行恢复了。

其次，AOF 虽然避免了对当前命令的阻塞，但可能会给下一个操作带来阻塞风险。这是因为，AOF 日志也是在主线程中执行的，如果在把日志文件写入磁盘时，磁盘写压力大，就会导致写盘很慢，进而导致后续的操作也无法执行了。

仔细分析的话，你就会发现，这两个风险都是和 AOF 写回磁盘的时机相关的。这也就意味着，如果我们能够控制一个写命令执行完后 AOF 日志写回磁盘的时机，这两个风险就解除了。

三种写回策略

其实，对于这个问题，AOF 机制给我们提供了三个选择，也就是 AOF 配置项 appendfsync 的三个可选值。

Always，同步写回：每个写命令执行完，立马同步地将日志写回磁盘；

Everysec，每秒写回：每个写命令执行完，只是先把日志写到 AOF 文件的内存缓冲区，每隔一秒把缓冲区中的内容写入磁盘；

No，操作系统控制的写回：每个写命令执行完，只是先把日志写到 AOF 文件的内存缓冲区，由操作系统决定何时将缓冲区内容写回磁盘。

针对避免主线程阻塞和减少数据丢失问题，这三种写回策略都无法做到两全其美。我们来分析下其中的原因。

“同步写回”可以做到基本不丢数据，但是它在每一个写命令后都有一个慢速的落盘操作，不可避免地会影响主线程性能；

虽然“操作系统控制的写回”在写完缓冲区后，就可以继续执行后续的命令，但是落盘的时机已经不在 Redis 手中了，只要 AOF 记录没有写回磁盘，一旦宕机对应的数据就丢失了；

“每秒写回”采用一秒写回一次的频率，避免了“同步写回”的性能开销，虽然减少了对系统性能的影响，但是如果发生宕机，上一秒内未落盘的命令操作仍然会丢失。所以，这只能算是，在避免影响主线程性能和避免数据丢失两者间取了个折中。

我把这三种策略的写回时机，以及优缺点汇总在了一张表格里，以方便你随时查看。

配置项	写回时机	优点	缺点
Always	同步写回	可靠性高，数据基本不丢失	每个写命令都要落盘，性能影响较大
Everysec	每秒写回	性能适中	宕机时丢失1秒内的数据
No	操作系统控制的写回	性能好	宕机时丢失数据较多

到这里，我们就可以根据系统对高性能和高可靠性的要求，来选择使用哪种写回策略了。总结一下就是：想要获得高性能，就选择 No 策略；如果想要得到高可靠性保证，就选择 Always 策略；如果允许数据有一点丢失，又希望性能别受太大影响的话，那么就选择 Everysec 策略。

但是，按照系统的性能需求选定了写回策略，并不是“高枕无忧”了。毕竟，AOF 是以文件的形式在记录接收到的所有写命令。随着接收的写命令越来越多，AOF 文件会越来越大。这也就意味着，我们一定要小心 AOF 文件过大带来的性能问题。

这里的“性能问题”，主要在于以下三个方面：一是，文件系统本身对文件大小有限制，无法保存过大的文件；二是，如果文件太大，之后再往里面追加命令记录的话，效率也会变低；三是，如果发生宕机，AOF 中记录的命令要一个个被重新执行，用于故障恢复，如果日志文件太大，整个恢复过程就会非常缓慢，这就会影响到 Redis 的正常使用。

所以，我们就要采取一定的控制手段，这个时候，**AOF 重写机制**就登场了。

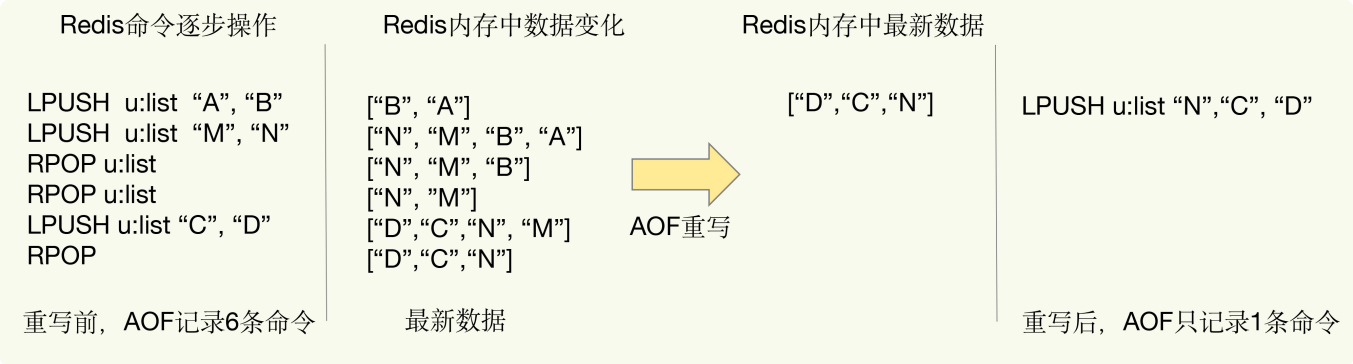
日志文件太大了怎么办？

简单来说，AOF 重写机制就是在重写时，Redis 根据数据库的现状创建一个新的 AOF 文件，也就是说，读取数据库中的所有键值对，然后对每一个键值对用一条命令记录它的写入。比如说，当读取了键值对“testkey”：“testvalue”之后，重写机制会记录 set testkey testvalue 这条命令。这样，当需要恢复时，可以重新执行该命令，实现“testkey”：“testvalue”的写入。

为什么重写机制可以把日志文件变小呢？实际上，重写机制具有“多变一”功能。所谓的“多变一”，也就是说，旧日志文件中的多条命令，在重写后的新日志中变成了一条命令。

我们知道，AOF 文件是以追加的方式，逐一记录接收到的写命令的。当一个键值对被多条写命令反复修改时，AOF 文件会记录相应的多条命令。但是，在重写的时候，是根据这个键值对当前的最新状态，为它生成对应的写入命令。这样一来，一个键值对在重写日志中只用一条命令就行了，而且，在日志恢复时，只用执行这条命令，就可以直接完成这个键值对的写入了。

下面这张图就是一个例子：



AOF重写减少日志大小

当我们对一个列表先后做了 6 次修改操作后，列表的最后状态是[“D”，“C”，“N”]，此时，只用 LPUSH u:list “N”，“C”，"D"这一条命令就能实现该数据的恢复，这就节省了五条命令的空间。对于被修改过成百上千次的键值对来说，重写能节省的空间当然就更大了。

不过，虽然 AOF 重写后，日志文件会缩小，但是，要把整个数据库的最新数据的操作日志都写回磁盘，仍然是一个非常耗时的过程。这时，我们就要继续关注另一个问题了：重写会不会阻塞主线程？

AOF 重写会阻塞吗？

和 AOF 日志由主线程写回不同，重写过程是由后台线程 bgrewriteaof 来完成的，这也是为了避免阻塞主线程，导致数据库性能下降。

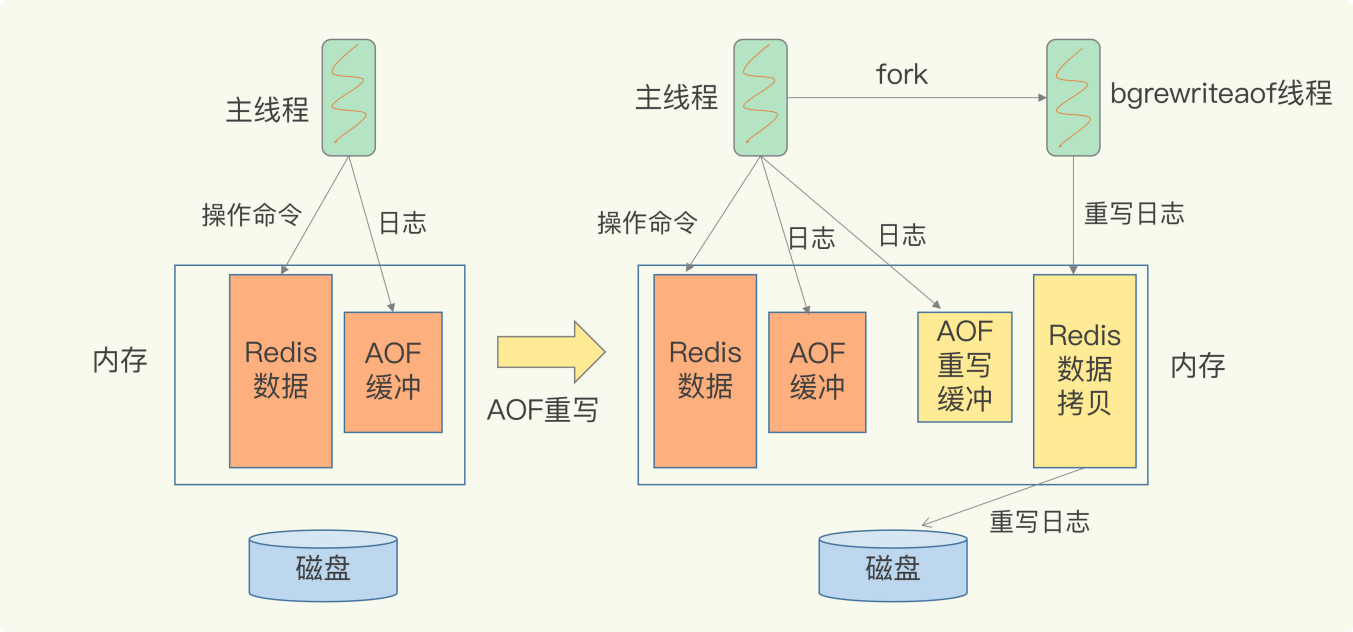
我把重写的过程总结为“**一个拷贝，两处日志**”。

“一个拷贝”就是指，每次执行重写时，主线程 fork 出后台的 bgrewriteaof 子进程。此时，fork 会把主线程的内存拷贝一份给 bgrewriteaof 子进程，这里面就包含了数据库的最新数据。然后，bgrewriteaof 子进程就可以在不影响主线程的情况下，逐一把拷贝的数据写成操作，记入重写日志。

“两处日志”又是什么呢？

因为主线程未阻塞，仍然可以处理新来的操作。此时，如果有写操作，第一处日志就是指正在使用的 AOF 日志，Redis 会把这个操作写到它的缓冲区。这样一来，即使宕机了，这个 AOF 日志的操作仍然是齐全的，可以用于恢复。

而第二处日志，就是指新的 AOF 重写日志。这个操作也会被写到重写日志的缓冲区。这样，重写日志也不会丢失最新的操作。等到拷贝数据的所有操作记录重写完成后，重写日志记录的这些最新操作也会写入新的 AOF 文件，以保证数据库最新状态的记录。此时，我们就可以用新的 AOF 文件替代旧文件了。



AOF非阻塞的重写过程

总结来说，每次 AOF 重写时，Redis 会先执行一个内存拷贝，用于重写；然后，使用两个日志保证在重写过程中，新写入的数据不会丢失。而且，因为 Redis 采用额外的线程进行数据重写，所以，这个过程并不会阻塞主线程。

小结

这节课，我向你介绍了 Redis 用于避免数据丢失的 AOF 方法。这个方法通过逐一记录操作命令，在恢复时再逐一执行命令的方式，保证了数据的可靠性。

这个方法看似“简单”，但也是充分考虑了对 Redis 性能的影响。总结来说，它提供了 AOF 日志的三种写回策略，分别是 Always、Everysec 和 No，这三种策略在可靠性上是从高到低，而在性能上则是从低到高。

此外，为了避免日志文件过大，Redis 还提供了 AOF 重写机制，直接根据数据库里数据的最新状态，生成这些数据的插入命令，作为新日志。这个过程通过后台线程完成，避免了对主线程的阻塞。

其中，三种写回策略体现了系统设计中的一个重要原则 ，即 trade-off，或者称为“取舍”，指的就是在性能和可靠性保证之间做取舍。我认为，这是做系统设计和开发的一个关键哲学，我也非常希望，你能充分地理解这个原则，并在日常开发中加以应用。

不过，你可能也注意到了，落盘时机和重写机制都是在“记日志”这一过程中发挥作用的。例如，落盘时机的选择可以避免记日志时阻塞主线程，重写可以避免日志文件过大。但是，在“用日志”的过程中，也就是使用 AOF 进行故障恢复时，我们仍然需要把所有的操作记录都运行一遍。再加上 Redis 的单线程设计，这些命令操作只能一条一条按顺序执行，这个“重放”的过程就会很慢了。

那么，有没有既能避免数据丢失，又能更快地恢复的方法呢？当然有，那就是 RDB 快照了。下节课，我们就一起学习一下，敬请期待。

每课一问

这节课，我给你提两个小问题：

1. AOF 日志重写的时候，是由 bgrewriteaof 子进程来完成的，不用主线程参与，我们今天说的非阻塞也是指子进程的执行不阻塞主线程。但是，你觉得，这个重写过程有没有其他潜在的阻塞风险呢？如果有的话，会在哪里阻塞？
2. AOF 重写也有一个重写日志，为什么它不共享使用 AOF 本身的日志呢？

希望你能好好思考一下这两个问题，欢迎在留言区分享你的答案。另外，也欢迎你把这节课的内容转发出去，和更多的人一起交流讨论。

提建议

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 03 | 高性能IO模型：为什么单线程Redis能那么快？

下一篇 05 | 内存快照：宕机后，Redis如何实现快速恢复？

精选留言 (42)

写留言

Kaito 置顶
2020-08-12

问题1，Redis采用fork子进程重写AOF文件时，潜在的阻塞风险包括：fork子进程 和 AOF重写过程中父进程产生写入的场景，下面依次介绍。

a、fork子进程，fork这个瞬间一定是会阻塞主线程的（注意，fork时并不会一次性拷贝所有内存数据给子进程，老师文章写的是拷贝所有内存数据给子进程，我个人认为是有...

展开

作者回复: 非常赞！这个回答一定要置顶！

而且Kaito同学的不少问题回答都非常仔细和精彩，非常值得一读！

这里要谢谢Kaito同学指出的文章中的歧义：fork子进程时，子进程是会拷贝父进程的页表，即虚实映射关系，而不会拷贝物理内存。子进程复制了父进程页表，也能共享访问父进程的内存数据了，此时，类似于有了父进程的所有内存数据。我的描述不太严谨了，非常感谢指出！

Kaito同学还提到了Huge page。这个特性大家在使用Redis也要注意。Huge page对提升TLB命中率比较友好，因为在相同的内存容量下，使用huge page可以减少页表项，TLB就可以缓存更多的页表项，能减少TLB miss的开销。

但是，这个机制对于Redis这种喜欢用fork的系统来说，的确不太友好，尤其是在Redis的写入请求比较多的情况下。因为fork后，父进程修改数据采用写时复制，复制的粒度为一个内存页。如果只是修改一个256B的数据，父进程需要读原来的内存页，然后再映射到新的物理地址写入。一读一写会造成读写放大。如果内存页越大（例如2MB的大页），那么读写放大也就越严重，对Redis性能造成影响。

Huge page在实际使用Redis时是建议关掉的。

21 80

徐鹏
2020-08-12

有几个问题想请教一下：
1、文中多处提到bgrewriteaof 子进程，这个有点迷糊，主线程fork出来的bgrewriteaof是子线程还是子进程？
2、AOF重写会拷贝一份完整的内存数据，这个会导致内存占用直接翻倍吗？
3、如果一个key设置了过期时间，在利用AOF文件恢复数据时，key已经过期了这个是...

展开

3 11

Darren
2020-08-12

AOF工作原理：

1、Redis 执行 fork() ，现在同时拥有父进程和子进程。
2、子进程开始将新 AOF 文件的内容写入到临时文件。
3、对于所有新执行的写入命令，父进程一边将它们累积到一个内存缓存中，一边将这些改动追加到现有 AOF 文件的末尾,这样样即使在重写的中途发生停机，现有的 AOF 文件也...
展开

4 7

d
2020-08-12

AOF 是什么的缩写， 还是说就是这个名字？
展开

作者回复: AOF的全称是Append Only File，表示文件只能追加写。 Redis记日志时，就是用追加写文件的方式记录写命令操作的。

3 3

曾轼麟
2020-08-12

问题一:在AOF重写期间，Redis运行的命令会被积累在缓冲区，待AOF重写结束后会进行回放，在高并发情况下缓冲区积累可能会很大，这样就会导致阻塞，Redis后来通过Linux管道技术让aof期间就能同时进行回放，这样aof重写结束后只需要回放少量剩余的数据即可
...
展开

1 3

注定非凡
2020-08-14

1，作者讲了什么？
本章讲了Redis两种持久化机制之一：AOF机制原理
aof日志记录了redis所有增删改的操作，保存在磁盘上，当redis宕机，需要恢复内存中的数据时，可以通过读取aop日志恢复数据，从而避免因redis异常导致的数据丢失
...
展开

2

杨逸林
2020-08-12

老师没有说什么时候会触发重写 ROF 文件，我 Google 一下， 重写的触发方式有下面两

个。

1. 手动执行 bgrewriteaof 触发 AOF 重写

...

展开

1 2

Anony

2020-08-12

上节说redis单线程处理网络IO和键值对的读写，持久化是由额外线程处理的。那AOF由额外线程处理的话，为什么会影响主线程呢？和主线程之间是什么关系？

展开

1 2

与路同飞

2020-08-12

1.子线程重新AOF日志完成时会向主线程发送信号处理函数，会完成（1）将AOF重写缓冲区的内容写入到新的AOF文件中。（2）将新的AOF文件改名，原子地替换现有的AOF文件。完成以后才会重新处理客户端请求。
2.不共享AOF本身的日志是防止锁竞争，类似于redis rehash。

展开

4 2

Monday

2020-08-15

老师好，本文循序渐进，线路清晰，实属深度好文，但如“AOF非阻塞的重写过程”图，有几个问题，望回复，谢谢。
1、如果写回策略是Always，是没有AOF缓冲的。是怎么做AOF重写呢？
2、图右边部分有四个日志：两个日志，两个重写日志，请问“一个拷贝，两个日志”中的“日志”是指四个中的哪两个？...

展开

1

Monday

2020-08-14

aof重写，我理解按如下操作也行：

bgrewriteaof线程直接操作磁盘的aof文件，主线程还是写原来的缓存。

1

一步
2020-08-14

appendfsync: no 是系统控制进行写磁盘，这里是利用文件系统的机制吗？ 如果是利用的文件系统的哪个功能， 一般多久进行持久化写磁盘呢？

展开

1

Archer30
2020-08-12

问题1回答：如果子进程写入事件过长，并且这段事件，会导致AOF重写日志，积累过多，当新的AOF文件完成后，还是需要写入大量AOF重写日志里的内容，可能会导致阻塞。

问题2回答：我觉得评论区里的大部分回答 防止锁竞争 ，应该是把问题理解错了，父子...

展开

1

脱缰的野马__
2020-08-12

文章前面说到redo log日志记录的是修改后的数据，但是在丁奇老师的MySQL实战中讲解的是redo log记录是对数据的操作记录，修改后的数据是保存在内存的change buffer中的

1

秋梵
2020-08-12

老师我有个问题想问。文章里说，在子进程进行重写期间，如果有新的写操作，Redis 会把这个操作写到它的缓冲区。但是缓冲区也是存在内存中的，如果宕机，内存的数据会丢失，新的写操作同样会丢失，这样的话要怎么保证AOF的日志操作是齐全的？

展开

1

test
2020-08-12

1.fork时候的开销
2.如果使用同一个文件会需要有文件锁的竞争

1

yyl
2020-08-12

提问：
1. 一个拷贝：拷贝是当前redis实例中所有的键值对吗？
2. "第一处日志就是指正在使用的 AOF 日志，Redis 会把这个操作写到它的缓冲区。这样一来，即使宕机了，这个 AOF 日志的操作仍然是齐全的，可以用于恢复。"
这里描述得不准确吧？操作记录写到缓存中，宕机也是会丢失的，正在使用的AOF日志...
展开

1 1

Mhy
2020-08-17

请问老师AOF日志为什么要记录GET操作，省下读操作的日志空间不好吗，如果是要判断access expire不能另辟蹊径吗
展开

Geek_Lin
2020-08-16

1.操作系统控制的写回：由操作系统决定何时将缓冲区内容写回磁盘，操作系统怎么决定的呢？
2.Redis是不是只有一个AOF文件，读文章感觉是，那么如果只有一个文章的话，追加写入操作，那么恢复的时候，如何知道从那一条操作开始呢？
展开

Geek_Lin
2020-08-16

每次都要把评论区看一遍，收获很大！！
展开